

Kartendarstellungen mit matplotlib Toolkit basemap

Simon von Hall

17. Dezember 2014

1 Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Seminararbeit mit dem Thema Kartendarstellungen mit `matplotlib Toolkit basemap` selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Name: Simon von Hall

Jülich, den 15.12.14

Unterschrift des Studenten

Inhaltsverzeichnis

1 Eidesstattliche Erklärung	2
2 Vorwort	5
3 Kartenprojektionen	5
3.1 Was sind Kartenprojektionen	5
3.2 Wofür braucht man Kartenprojektionen	5
3.3 Wie unterscheiden sich die Projektionen	5
3.4 Welche Kartenprojektionen unterstützt Basemap	6
4 Basemap	9
4.1 Einführung	9
4.2 Abhängigkeiten von basemap	10
4.3 Erstellen einer Karte mit Basemap	11
4.4 Zeichnen der Karte	14
4.5 Bild auf eine Karte zeichnen	22
4.6 Daten plotten	24
4.6.1 Isobaren plotten	24
4.6.2 Windmarken plotten	27
4.6.3 Windvektoren plotten	28
4.6.4 gekrümmte Strecken plotten	29
4.6.5 Viele Punkte plotten	30
4.6.6 Linienzüge zeichnen	31
4.7 Karte speichern	31
4.8 Karte anzeigen	31
5 Anhang	35
5.1 Azimuthale äquidistante Projektion	35
5.2 Gnomonische Projektion	36
5.3 Orthographische Projektion	37
5.4 Geostationäre Projektion	38
5.5 Nah-seitige perspektivische Projektion	39
5.6 Mollweide Projektion	40
5.7 Hammer Projektion	41
5.8 Robinson Projektion	42
5.9 Eckert 4 Projektion	43
5.10 Kavrayskiy 7 Projektion	44
5.11 McBryde-Thomas Projektion	45
5.12 Sinus-förmige Projektion	46
5.13 Äquidistante Zylinder Projektion	47
5.14 Cassini Projektion	48
5.15 Mercatorprojektion	49

5.16 Transversale Mercatorprojektion	50
5.17 Schiefe Mercatorprojektion	51
5.18 Polykonische Projektion	52
5.19 Miller Zylinderprojektion	53
5.20 Stereographische Gall Projektion	54
5.21 Flächentreue Zylinderprojektion	55
5.22 Winkeltreue Lambert Projektion	56
5.23 Azimuthale Flächentreue Lambert-Projektion	57
5.24 Stereografische Projektion	58
5.25 Längentreue Kegelprojektion	59
5.26 Flächentreue Albert-Projektion	60
5.27 Polare stereographische Projektion	61
5.28 Polare azimutale Lambertprojektion	62
5.29 Polare azimuthale äquidistante Projektion	64
5.30 Van der Grinten Projektion	65

2 Vorwort

In dieser Seminararbeit möchte ich das Python Modul Basemap vorstellen. Dazu werde ich die verschiedenen Kartenprojektionen die es unterstützt kurz beschreiben. Danach gehe ich auf die Funktionen des Moduls ein und beschreibe was diese machen. Dabei gehe ich dann noch ganz kurz darauf ein was für Voraussetzungen das Modul erwartet. Ich erkläre außerdem kurz wie man einfach mit dem Koordinatensystem umgehen kann. Zum Schluss erkläre ich noch ganz kurz wie man die Karten die man erstellt hat speichert.

3 Kartenprojektionen

In diesem Kapitel werde ich darauf eingehen was Kartenprojektionen sind, wofür man sie braucht und wie sie sich grundlegend unterscheiden.

3.1 Was sind Kartenprojektionen

Kartenprojektionen sind 2D Darstellungen der Erde oder spezieller Regionen der Erde. Dabei ist wichtig dass man die Landflächen von den Gewässern unterscheiden kann. Die Projektionen werden gebraucht wenn man geographische Daten darstellen will.

3.2 Wofür braucht man Kartenprojektionen

Kartenprojektionen braucht man wenn man ein 3D Objekt wie die Erde zweidimensional darstellen muss. Die Projektionen ermöglichen es die Punkte im Raum auf Punkte in der Ebene zu projizieren. Dabei gibt es verschiedene Varianten die unterschiedliche Vor- und Nachteile haben. Um geografische Daten wie zum Beispiel der Niederschlag auf ein Satellitenbild zu malen braucht man die entsprechende Projektion. Damit man die geografischen Koordinaten in die kartesischen Koordinaten des Bildes umrechnen kann.

3.3 Wie unterscheiden sich die Projektionen

Da die Projektionen die Dimension der Darstellung reduzieren, kann man nicht alles verzerrungsfrei darstellen. Die Projektionen unterscheiden sich einmal darin welche Verzerrung sie minimieren und zum anderen wie sie die Projektion erreichen. Es gibt zum einen flächentreue Projektionen. Diese stellen sicher dass die Flächen nicht verzerrt werden, dies bedeutet nicht dass die Form der Gebiete erhalten bleibt. Andererseits gibt es winkeltreue Projektionen diese stellen sicher dass die Winkel zwischen zwei Punkten erhalten bleiben. Dies wiederum bedeutet nicht dass die Entfernung zwischen zwei Punkten gleich bleibt wenn man diese parallel verschiebt noch dass die Flächen

gleich bleiben. Dann gibt es noch längentreue Projektionen diese stellen sicher das Entfernung erhalten bleiben wenn man diese verschiebt. Zuletzt gibt es noch Projektionen die einen Kompromiss aus den oben genannten Eigenschaften bilden.

Außerdem kann man Projektionen dadurch unterscheiden wie sie projizieren. Es gibt zum einen Zylinderprojektionen diese projizieren im Grunde indem sie einen Zylinder um die Erde legen und dann die Punkte linear übertragen, der Zylindermantel ist dann in der Regel die Karte. Des weiteren gibt es Kegelprojektionen diese bauen einen Kegel auf. Dieser wird in der Regel so aufgebaut das er tangential an der Erde aufliegt. Dann werden die Punkte auf den Kegelmantel übertragen. Dabei wird als Ausgangslinie häufig ein Breitenkreis genommen in diesem Fall laufen dann die Längengrade in der Spitze des Kegels zusammen. Die Karte erhält man dann wenn man den Kegel Aufschneidet und ausrollt. Dann gibt es noch die azimutale Projektion oder planare Projektion diese werden hauptsächlich durch ihr Projektionszentrum bestimmt. Dieser liegt bei gnomonischen Projektionen liegt dieser im Mittelpunkt der Erde, bei stereografischen Projektionen liegt er auf der Erdoberfläche und bei orthografischen Projektionen liegt er unendlich weit entfernt. Die azimutale Projektion wird gebildet, in dem vom Projektionszentrum Geraden zur Projektionsebene gebildet werden. Diese Ebene liegt normalerweise auf der Seite der Erde die dem Projektionszentrum gegenüber liegt. Die Punkte werden dann einfach entlang der gebildeten geraden auf die Ebene verschoben. Dabei werden die meisten geraden Sekanten sein was bedeutet das sie die Erdoberfläche zweimal durchstoßen. Des wegen unterscheidet man noch die Nah-seitige und die Fern-seitige Projektion. Bei der Nah-seitigen Projektion werden die ersten Schnittpunkte auf die Projektionsebene übertragen. Bei der Fern-seitigen Projektion werden entsprechend die zweiten Schnittpunkte auf die Projektionsebene übertragen. Zuletzt gibt es noch Projektionen, die die Koordinaten einfach mittel mathematischer Formeln umwandeln. Diese müssen nicht unbedingt geometrisch beschreibbar sein.

3.4 Welche Kartenprojektionen unterstützt Basemap

Das Basemap Modul unterstützt mehrere Projektionen. Die unterstützten Projektionen beschreibe ich kurz im Anhang. Die Folgenden Projektionen werden unterstützt:

- Azimuthale äquidistante Projektion
- Gnomonische Projektion

- Orthographische Projektion
- Geostationäre Projektion
- Nah-seitige perspektivische Projektion
- Mollweiden Projektion
- Hammer Projektion
- Robinson Projektion
- Eckert 4 Projektion
- Kavrayskiy 7 Projektion
- McBryde-Thomas Projektion
- Sinus-förmige Projektion
- Äquidistante Zylinderprojektion
- Cassini Projektion
- Mercatorprojektion
- Transversale Mercatorprojektion
- Schiefe Mercatorprojektion
- Polykonische Projektion
- Miller Zylinderprojektion

- Stereographische Gall Projektion
- Flächentreue Zylinderprojektion
- Winkeltreue Lambert Projektion
- Azimuthale Flächentreue Lambert-Projektion
- Stereographische Projektion
- Längentreue Kegelprojektion
- Flächentreue Albert-Projektion
- Polare stereographische Projektion
- Polare azimutale Lambertprojektion
- Polare azimuthale äquidistante Projektion
- Van der Grinten Projektion

Abhängig von der Anwendung sollte man eine dieser Projektionen wählen.
Mit diesen Projektionen sollten die meisten Probleme lösbar sein.

4 Basemap

4.1 Einführung

Das Toolkit *Basemap* ist ein Pythonmodul zur Bearbeitung von Karten. Es bietet viele verschiedene Arten Karten in 2D darzustellen (siehe Projektionen ??) Es ermöglicht einem auch das Plotten auf den Karten. Hierbei kann man, dann auch Längen- und Breitengrad als Positionsangabe nutzen. Das *Basemap* Toolkit wandelt die Koordinaten dann mit der *PROJ4* Library in die entsprechenden 2D Koordinaten um. Beim Plotten greift das Modul auf Funktionen des Moduls *pyplot* und *matplotlib* zurück. Der große Vorteil beim benutzen des *Basemap* Moduls ist, dass es die geografischen Koordinaten sehr einfach umrechnet. Was einem die Definition der Projektionsalgorithmen abnimmt. Des weiteren hat dieses Modul sehr präzise Umrissdaten, was es ermöglicht die Länder sehr genau darzustellen. Es bietet auch gröbere Umrissdaten, wenn man nicht so viel Rechenzeit hat. Mit diesem Modul ist es äußerst einfach, meteorologische Karten zu erstellen. Da es viele Funktionen bereitstellt mit denen man die Daten direkt darstellen kann. Es gibt Funktionen, um Windmarken oder Vektoren zu zeichnen. Andere Funktionen zeichnen Isobaren oder Strömungsgraphen oder farbig gefüllte Isobaren, diese können einfach zum Darstellen von Niederschlag verwendet werden. Diese Funktionen müssen einfach nur mit den Daten aufgerufen werden, diese Daten können auch als Felder vorliegen. Da das Modul das umrechnen der Koordinaten sehr einfach macht, können die Daten meistens einfach aus den Eingabedateien in die Felder gelesen werden, dies ist in Python sehr leicht. In diesem Kapitel werde ich die Funktionen von *Basemap* an einem Beispiel vorstellen. Bei diesem Beispiel werde ich die äquidistante Zylinderprojektion benutzen. Diese Projektion verwende ich, da sie die einfachste Projektion ist. Ich werde dabei darauf achten das die Beispiele auch mit anderen Projektionen funktionieren könnten.

Hier ein Beispiel zum umrechnen von Koordinaten

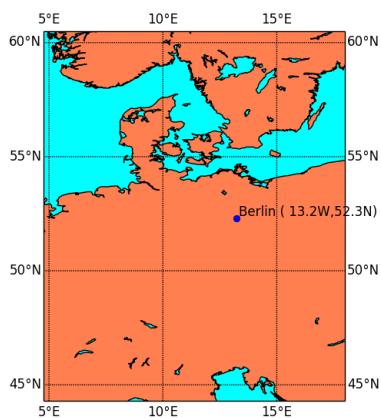
```
1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n                 llcrnrlon=4.750,urcrnrlon=18.00)
6 m.drawcoastlines()
7 m.fillcontinents(color='coral', lake_color='aqua')
# draw parallels and meridians.
8 m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
9             False])
10 m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
11             True, True])
12 m.drawmapboundary(fill_color='aqua')
# plt.title("Equidistant Cylindrical Projection")
13 #plt.show()
```

```

15 lon, lat = 13.2437, 52.3127 # Location of Berlin
# convert to map projection coords.
# Note that lon,lat can be scalars, lists or numpy arrays.
17 xpt,ypt = m(lon,lat)
# convert back to lat/lon
19 lonpt, latpt = m(xpt,ypt,inverse=True)
m.plot(xpt,ypt,'bo') # plot a blue dot there
21 # put some text next to the dot, offset a little bit
# (the offset is in map projection coordinates)
23 plt.text(xpt+0.1,ypt+0.1,'Berlin (%5.1fW,%3.1fN)' % (lonpt,latpt
    ))
plt.savefig('/home/s.von.hall/seminar/bspconvert.png')

```

/Users/student/seminar/bsp/bspconvert.py



4.2 Abhängigkeiten von basemap

Das Toolkit setzt folgende Module voraus:

- Python
die Skriptsprache des Moduls
- Numpy
Numpy ist ein Modul für numerische Berechnungen
- Matplotlib
Matplotlib ist das Plottingmodul das Basemap erweitert
- GEOS (ist im Projekt enthalten)
GEOS ist eine Geometrie-Engine
- PROJ4 (ist im Projekt enthalten)
PROJ4 ist eine kartografische Projektions Library

- PIL (ist optional)
PIL wird nur gebraucht um auf mehr Bildformate zugreifen zu können.

Links zu den Downloadseiten sind in der Basemap Dokumentation zu finden.

4.3 Erstellen einer Karte mit Basemap

Eine Karte erzeugt man mit der Klasse *Basemap*. Die Funktion ist wie folgt definiert:

```
Basemap(lcrnrlon=None, lcrnrlat=None, urcrnrlon=None, urcrnrlat=None, lcrnrx=None, lcrnry=None, urcrnrx=None, urcrnry=None, width=None, height=None, projection='cyl', resolution='c', area_thresh=None, rsphere=6370997.0, ellps=None, lat_ts=None, lat_1=None, lat_2=None, lat_0=None, lon_0=None, lon_1=None, lon_2=None, o_lon_p=None, o_lat_p=None, k_0=None, no_rot=False, suppress_ticks=True, satellite_height=35786000, boundinglat=None, fix_aspect=True, anchor='C', celestial=False, round=False, epsg=None, ax=None)
```

Dabei kann der Parameter `projection` folgende Werte haben:

- cea Cylindrical Equal Area
- mbtfpq McBryde-Thomas Flat-Polar Quartic
- aeqd Azimuthal Equidistant
- sinu Sinusoidal
- poly Polyconic
- omerc Oblique Mercator
- gnom Gnomonic
- moll Mollweide
- lcc Lambert Conformal
- tmerc Transverse Mercator
- nplaea North-Polar Lambert Azimuthal
- gall Gall Stereographic Cylindrical
- nphaeqd North-Polar Azimuthal Equidistant
- mill Miller Cylindrical
- merc Mercator
- stere Stereographic
- eqdc Equidistant Conic
- rotpole Rotated Pole
- cyl Cylindrical Equidistant
- npstere North-Polar Stereographic
- spstere South-Polar Stereographic
- hammer Hammer
 - geos Geostationary
 - nsper Near-Sided Perspective
 - eck4 Eckert IV
 - aea Albers Equal Area
 - kav7 Kavrayskiy VII
 - spaeqd South-Polar Azimuthal Equidistant
 - nphaepd North-Polar Azimuthal Equidistant
 - ortho Orthographic
 - cass Cassini-Soldner
 - vandg van der Grinten
 - laea Lambert Azimuthal Equal Area
 - splaea South-Polar Lambert Azimuthal
 - nplaea North-Polar Lambert Azimuthal
 - robin Robinson

Die Parameter `width`, `height` geben die Breite, beziehungsweise die Höhe der Karte in Metern an, diese Parameter können bei den folgenden Projektionen nicht gesetzt werden.

`sinu`, `moll`, `hammer`, `npstere`, `spstere`, `nplaea`, `splaea`, `nphaepd`, `spaeqd`,
`robin`, `eck4`, `kav7`, `mbtfpq`, `ortho`, `geos`, `nsper`

Die Parameter `lon_0`, `lat_0` nehmen die Koordinate des Kartenmittelpunkts in Grad, beziehungsweise den zentralen Längen- oder Breitengrad.

Die Parameter `urcrnrlon`, `urcrnrlat`, `llcrnrlon`, `llcrnrlat`, `urcrnrx`, `urcrnry`, `llcrnrx`, `llcrnry` sind die Koordinaten der untere linke und oberen rechten Ecke der Karte, in Grad oder Meter mit (0,0) im Mittelpunkt der Karte. Eine der beiden Arten die Grenzen der Karte festzulegen muss angegeben werden außer bei den folgenden Projektionen:

```
sinu, moll, hammer, npstere, spstere, nplaea, splaea, npaepd, spaepd,  
robin, eck4, kav7, mbtfpq
```

Da diese entweder immer den ganzen Globus zeigen oder die Grenzen automatisch ermittelt werden. Bei der `rotpole` Projektion werden mit `lat/lon` die Ecken des nicht rotierten Globus angegeben mit `x/y` die Ecken des rotierten Globus. Bei dem Parameter `resolution` kann man die Werte `c`, `l`, `i`, `h`, `f`, `None` haben, falls `None` angegeben wurde, werden keine Grenzdaten geladen weshalb Klassenmethoden Fehler werfen.

Der Parameter `area_tresh` gibt an welche Mindest-Fläche Seen haben müssen um gezeichnet zu werden. Der Defaultwert ist 10000, 1000, 100, 10, 1 für die Resolution von `c`, `l`, `i`, `h`, `f`. Die Werte geben die Fläche in km^2 .

Der Parameter `rsphere` bekommt den Radius, der der Projektion zugrunde liegenden Figur, in Metern. Bei einer Kugel wird nur ein Wert angegeben, bei einem Ellipsoiden wird ein Array mit zwei Werten erwartet.

Über den Parameter `ellps` kann man die Form der Erde mit Hilfe spezieller Zeichenketten angeben. Sollte `ellps` angegeben sei wird `rsphere` ignoriert.

Mit dem Parameter `supress_ticks` kann man steuern ob die Achsen automatisch beschriftet und aufgeteilt werden sollen.

Der Parameter `fix_aspect` passt die Seitenverhältnisse vom Plot den Seitenverhältnissen der Karte an.

Der Parameter `anchor` gibt an wo die Karte im Gitter liegt, gültige Werte für diesen Parameter sind:

```
C, SW, S, SE, E, NE, N, NW, W
```

Mit dem Parameter `celestial` kann man einstellen das eine astronomische Konvention bezüglich der Längengrade benutzt wird. Was bedeutet, dass negative Längengrade östlich des Nullmeridian liegen. Diese Einstellung impliziert das die Resolution auf `None` gesetzt ist.

Mit dem `ax` Parameter kann man eine eigene Achseninstanz übergeben, wenn nichts übergeben wird, versucht `basemap` sich die aktuelle Achseninstanz mit Hilfe von `pyplot.gca()` zu holen. Falls man nicht `pyplot` importiert kann man auch jeder Klassenmethode die zeichnet eine Achseninstanz übergeben auf der gezeichnet wird. Wenn man allerdings hier die Achseninstanz übergibt

werden alle Methoden auf dieser Achseninstanz ausgeführt.

Mit dem `lat_ts` Parameter wird der Breitengrad angegeben, der Maßstabsgetreu wiedergegeben wird. Dieser Parameter ist optional und nur für die Projektionen `stere`, `cea`, `merc` von Belang.

Mit den Parametern `lat_1`, `lat_2` werden die Breitengrade der 1. und 2. Standardparallele angegeben. Diese sind nur für die Projektionen `lcc`, `aea`, `eqdc` interessant.

Mit den Parametern `lon_1`, `lon_2` werden die Breitengerade für die Punkte auf der Zentralen Linie der `omerc` Projektion angegeben.

Der Parameter `k_0` gibt den Skalierungsfaktor am Ursprung an, welcher nur von den Projektionen `tmerc`, `omerc`, `stere`, `lcc` genutzt wird.

Mit dem Parameter `no_rot` kann man bei der `omerc` Projektion einstellen ob die Koordinaten rotiert werden oder nicht. Die Parameter `o_lat_p`, `o_lon_p` geben die Position des rotierten Pols in Grad an, dies wird nur von der Projektion `rotpole` verarbeitet.

Der Parameter `boundinglat` gibt an bis zu welchem Breitengrad die polare Karte reicht.

Mit dem Parameter `round` kann man einstellen ob an dem Grenzbreitengrad bei polaren Karten abgeschnitten werden soll oder nicht. Dadurch kann man einstellen ob die Karte rund oder eckig ist.

Mit dem Parameter `satellite_height` gibt man die Höhe des Satelliten über dem Äquator an. Dies ist nur für die Projektionen `geos`, `nsper` interessant.

Mit dieser Funktion kann man eine Karteninstanz erstellen, allerdings ist noch nichts gezeichnet worden. Dafür müssen noch weitere Funktionen aufgerufen werden. Welche ich im nächsten Unterkapitel beschreibe.

4.4 Zeichnen der Karte

Die Karte zeichnet man mit verschiedenen Methoden der Karte.

Küstenlinie zeichnen

Die Küstenlinie zeichnet man mit der Funktion `drawcoastlines(linewidth=1.0, linestyle='solid', color='k', antialiased=1, ax=None, zorder=None)`, man kann mit den Parametern einstellen wie die Küstenlinie gezeichnet werden soll. Mit dem Parameter `zorder` kann man die Reihenfolge in der gezeichnet wird beeinflussen. Niedrige Werte im Parameter `zorder` werden zuerst gezeichnet. Mit dem Parameter `ax` kann man eine Achseninstanz übergeben auf der gezeichnet werden soll, normalerweise wird die Achseninstanz der Karte genommen. Sollte keine Achseninstanz gefunden werden wird ein Fehler geworfen.

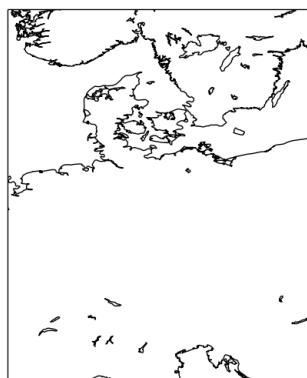
```
from mpl_toolkits.basemap import Basemap
```

```

2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n
6             llcrnrlon=4.750,urcrnrlon=18.00)
7 m.drawcoastlines()
8 #m.fillcontinents(color='coral', lake_color='aqua')
9 # draw parallels and meridians.
10 #m.drawparallels(np.arange(-90.,91.,5))#, labels=[True, True,
11 #           True, False])
12 #m.drawmeridians(np.arange(-180.,181.,5))#, labels=[True, False,
13 #           True, True])
14 #m.drawmapboundary(fill_color='aqua')
15 #plt.title("Equidistant Cylindrical Projection")
16 #plt.show()
17 plt.savefig('/home/s.von.hall/seminar/bspcoast.png')

```

/Users/student/seminar/bsp/bspcoast.py

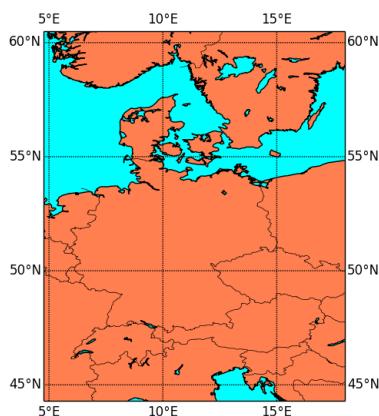


Ländergrenzen zeichnen

Ländergrenzen können mit der Funktion `drawcountries(linewidth=0.5, linestyle='solid', color='k', antialiased=1, ax=None, zorder=None)` gezeichnet werden. Die Parameter sind die selben wie beim zeichnen der Küstenlinien. Mit der Funktion `drawcountries(linewidth=0.1, linestyle='solid', color='k', antialiased=1, facecolor='none', ax=None, zorder=None, drawbounds=False)` können die Countiegrenzen in den USA gezeichnet werden. Mit dem Parameter `facecolor` kann eine Farbe angegeben werden mit der gefüllt werden soll. Mit der Funktion `drawstates(linewidth=0.5, linestyle='solid', color='k', antialiased=1, ax=None, zorder=None)` können die Staatsgrenzen in den USA gezeichnet werden.

```
from mpl_toolkits.basemap import Basemap
2 import numpy as np
import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
urcrnrlat=60.5,\n            llcrnrlon=4.750,urcrnrlon=18.00)
6 m.drawcoastlines()
m.fillcontinents(color='coral', lake_color='aqua')
8 # draw parallels and meridians.
m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
False])
10 m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
True, True])
m.drawmapboundary(fill_color='aqua')
12 m.drawcountries()
# plt.title("Equidistant Cylindrical Projection")
14 # plt.show()
plt.savefig('/home/s.von.hall/seminar/bspland.png')
```

/Users/student/seminar/bsp/bspland.py



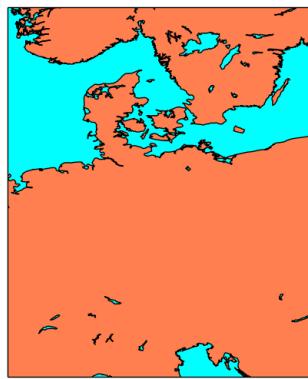
Fülle der Karte

Mit der Funktion `fillcontinents(color='0.8', lake_color=None, ax=None, zorder=None, alpha=None)` kann man die Kontinente und Seen mit Farben füllen. Mit der Funktion `drawmapboundary(color='k', linewidth=1.0, fill_color=None, zorder=None, ax=None)` kann man die Kartengrenze zeichnen und die Karte mit einer Farbe füllen. So kann man die Ozeane mit einer Farbe füllen, wenn man die Kontinente ebenfalls mit Farbe füllt. Sonst hat alles die gleiche Farbe.

Man kann die Karte auch mit Hilfe der Funktion `drawlsmask(land_color='0.8', ocean_color='w', lsmask=None, lsmask_lons=None, lsmask_lats=None, lakes=True, resolution='l', grid=5, **kwargs)` füllen. Diese Funktion zeichnet ein Bild, daher kann man keine Zeichenreihenfolge angeben. Die Funktion erwartet im Parameter `lsmask` eine Matrix mit den Werten 0 für ein Ozean Pixel, 1 für ein Land Pixel und 2 für ein See Pixel. In den Parametern `lsmask_lon` und `lsmask_lat` erwartet die Funktion eindimensionale Arrays für die Längen- und Breitengrade der `lsmask`, sie müssen aufsteigend sortiert sein.

```
1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n                 llcrnrlon=4.750,urcrnrlon=18.00)
6 m.drawcoastlines()
7 m.fillcontinents(color='coral',lake_color='aqua')
8 # draw parallels and meridians.
9 #m.drawparallels(np.arange(-90.,91.,5))#, labels=[True, True,
10 #              True, False])
11 #m.drawmeridians(np.arange(-180.,181.,5))#, labels=[True, False,
12 #               True, True])
13 m.drawmapboundary(fill_color='aqua')
14 #plt.title("Equidistant Cylindrical Projection")
15 #plt.show()
16 plt.savefig('/home/s.von.hall/seminar/bspbase.png')
```

/Users/student/seminar/bsp/bspbase.py

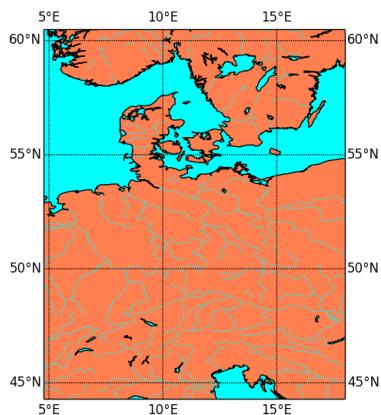


Flüsse zeichnen

Die Funktion `drawrivers(linewidth=0.5, linestyle='solid', color='k', antialiased=1, ax=None, zorder=None)` ermöglicht es Flüsse zu zeichnen. Es sind allerdings nicht alle Flüsse erfasst. Die größeren sind allerdings erfasst.

```
from mpl_toolkits.basemap import Basemap
2 import numpy as np
import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
      urcrnrlat=60.5,\n           llcrnrlon=4.750,urcrnrlon=18.00)
6 m.drawcoastlines()
m.fillcontinents(color='coral', lake_color='aqua')
8 # draw parallels and meridians.
m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
      False])
10 m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
      True, True])
m.drawmapboundary(fill_color='aqua')
12 m.drawrivers(color='aqua')
# plt.title("Equidistant Cylindrical Projection")
14 #plt.show()
plt.savefig('/home/s.von.hall/seminar/bsprivers.png')
```

/Users/student/seminar/bsp/bsprivers.py

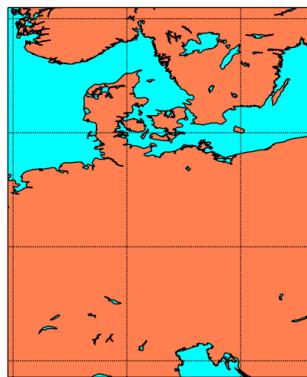


Längen- und Breitengrade zeichnen

Längen- und Breitengrade kann man einfach mit den Funktionen `drawmeridians(meridians, color='k', linewidth=1.0, zorder=None, dashes=[1, 1], labels=[0, 0, 0, 0], labelstyle=None, fmt='%g', xoffset=None, yoffset=None, ax=None, latmax=None, **kwargs)` und `drawparallels(circles, color='k', linewidth=1.0, zorder=None, dashes=[1, 1], labels=[0, 0, 0, 0], labelstyle=None, fmt='%g', xoffset=None, yoffset=None, ax=None, latmax=None, **kwargs)` zeichnen. Der Parameter `labels` gibt an wo die Grade beschriftet werden sollen links, rechts, oben, unten. Mit dem Parameter `labelstyle` kann man einstellen ob die Grade mit \pm oder mit E, W beziehungsweise N, S beschriftet werden sollen. Falls nichts angegeben wird, werden die Buchstaben benutzt. Dem Parameter `fmt` kann eine Funktion übergeben werden die aus dem Gradwert eine Zeichenkette macht. Mit dem Parameter `meridians` beziehungsweise `circles` wird eine Liste der Grade übergeben die gezeichnet werden sollen. Über den Parameter `dashes` kann man ein Pattern definieren mit dem die Grade gezeichnet werden sollen. Es werden immer abwechselnd die Anzahl an Pixeln angegeben, die gezeichnet und nicht gezeichnet werden sollen.

```
1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n
6                 llcrnrlon=4.750,urcrnrlon=18.00)
7 m.drawcoastlines()
8 m.fillcontinents(color='coral',lake_color='aqua')
# draw parallels and meridians.
9 m.drawparallels(np.arange(-90.,91.,5))#, labels=[True, True,
10                 True, False])
11 m.drawmeridians(np.arange(-180.,181.,5))#, labels=[True, False,
12                 True, True])
13 m.drawmapboundary(fill_color='aqua')
# plt.title("Equidistant Cylindrical Projection")
# plt.show()
plt.savefig('/home/s.von.hall/seminar/bspgrade.png')
```

/Users/student/seminar/bsp/bspgrad.py



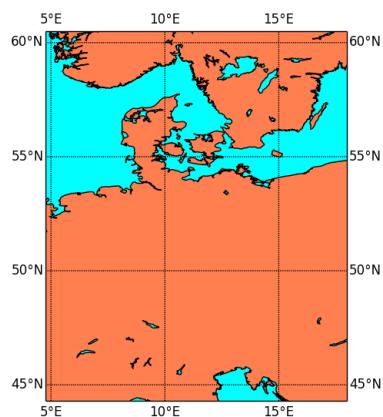
Mit Beschriftungen:

```

1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n
6             llcrnrlon=4.750,urcrnrlon=18.00)
7 m.drawcoastlines()
8 m.fillcontinents(color='coral', lake_color='aqua')
9 # draw parallels and meridians.
10 m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
11         False])
12 m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
13         True, True])
14 m.drawmapboundary(fill_color='aqua')
15 #plt.title("Equidistant Cylindrical Projection")
16 #plt.show()
17 plt.savefig('/home/s.von.hall/seminar/bspgradlabel.png')

```

/Users/student/seminar/bsp/bspgradlabel.py

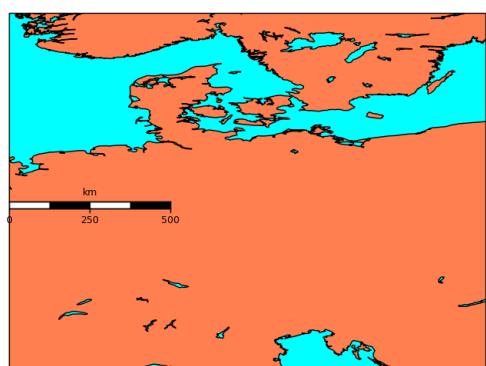


Maßstab zeichnen

Um einen Maßstab zu zeichnen braucht man die Funktion `drawmapscale(lon, lat, lon0, lat0, length, barstyle='simple', units='km', fontsize=9, yoffset=None, labelstyle='simple', fontcolor='k', fillcolor1='w', fillcolor2='k', ax=None, format='%.d', zorder=None)`. Diese Funktion kann nicht mit der Projektion cyl ausgeführt werden. Die Funktion zeichnet eine Skala an der Position lon, lat der Länge length von dem Punkt lon0, lat0. Die beiden Positionen sind wichtig da die Projektion längenverzerrend sein kann. Man kann den `barstyle` auf `fancy` stellen, dann wird eine Skala gezeichnet bei der sich verschieden farbige Balken abwechseln. Wenn beim Parameter `labelstyle fancy` angegeben wird, wird über dem Balken noch der Verzerrungsfaktor und die Position lon0, lat0 ausgegeben. Mit dem Parameter `yoffset` kann man die Höhe der Skala und die Entfernung der Beschriftung vom Balken in Metern angeben. Der Defaultwert hierfür ist 2% der Karten Höhe.

```
from mpl_toolkits.basemap import Basemap
2 import numpy as np
import matplotlib.pyplot as plt
4 m = Basemap(projection='cea', resolution='i', llcrnrlat=44.30,
      urcrnrlat=60.5,\n           llcrnrlon=4.750,urcrnrlon=18.00)
6 m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
8 # draw parallels and meridians.
#m.drawparallels(np.arange(-90.,91.,5))#, labels=[True, True,
      True, False])
10 #m.drawmeridians(np.arange(-180.,181.,5))#, labels=[True, False,
      True, True])
m.drawmapboundary(fill_color='aqua')
12 x, y=7, 51
m.drawmapscale( x, y, x, y,500, barstyle='fancy' )
14 #plt.title("Equidistant Cylindrical Projection")
#plt.show()
16 plt.savefig('/home/s.von.hall/seminar/bspscalar.png')
```

/Users/student/seminar/bsp/bspscalar.py



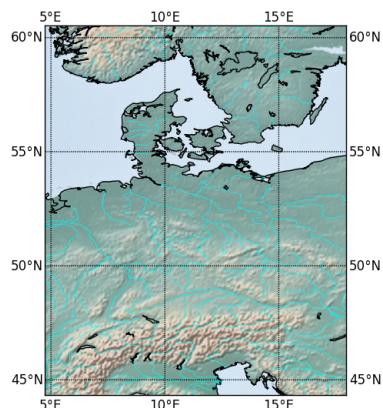
Relief zeichnen

Um eine Karte mit Relief zu zeichnen kann man die Funktionen `etopo(ax=None, scale=None, **kwargs)` oder `shadedrelief(ax=None, scale=None, **kwargs)` benutzen. Die Funktion `etopo` lädt ein ein Reliefbild von der Seite <http://www.ngdc.noaa.gov/mgg/global/> als Hintergrund. Die Funktion `shadedrelief` lädt ein Reliefbild von der Seite <http://www.shadedrelief.com> als Hintergrund. Mit dem Parameter `scale` kann man einen Skalierungsfaktor angeben, da die Bilder 10800x5400 groß sind. Was eine gewisse Zeit zum laden und verarbeiten braucht und Speicherplatz verbraucht.

Mit `shadedrelief()`:

```
from mpl_toolkits.basemap import Basemap
2 import numpy as np
import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
      urcrnrlat=60.5,\n        llcrnrlon=4.750,urcrnrlon=18.00)
6 m.drawcoastlines()
m.shadedrelief()
8 #m.fillcontinents(color='coral', lake_color='aqua')
# draw parallels and meridians.
10 m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
       False])
m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False, True,
       True])
12 #m.drawmapboundary(fill_color='aqua')
m.drawrivers(color='aqua')
14
# plt.title("Equidistant Cylindrical Projection")
16 #plt.show()
plt.savefig('/home/s.von.hall/seminar/bspshadereliev.png')
```

/Users/student/seminar/bsp/bspshadereliev.py



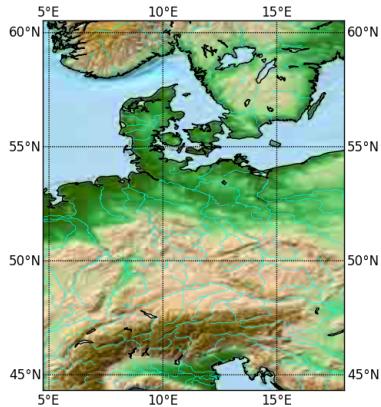
Mit `etopo()`:

```

1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n
6                 llcrnrlon=4.750,urcrnrlon=18.00)
7 m.drawcoastlines()
8 m.etopo()
#m.fillcontinents(color='coral', lake_color='aqua')
9 # draw parallels and meridians.
10 m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
11 False])
12 m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False, True,
13 True])
#m.drawmapboundary(fill_color='aqua')
14 m.drawrivers(color='aqua')
15 #plt.title("Equidistant Cylindrical Projection")
16 #plt.show()
17 plt.savefig('/home/s.von.hall/seminar/bspetopo.png')

```

/Users/student/seminar/bsp/bspetopo.py



4.5 Bild auf eine Karte zeichnen

Um ein Bild auf eine Karte zeichnen zu können, muss das Bild als `pyplot.image` vorliegen. Dies kann man mit Hilfe der `PIL` erreichen. Da `matplotlib.image` eine Funktion `pil_to_array(Pilimage)` zur Verfügung stellt. Beim Zeichnen von Bildern muss man beachten das sie über die ganze Karte gezeichnet werden. Die Funktion um ein Bild zu zeichnen ist `imshow(X, cmap=None, norm=None, aspect=None, interpolation=None, alpha=None, vmin=None, vmax=None, origin=None, extent=None, shape=None, filternorm=1, filterrad=4.0, imlim=None, resample=None, url=None, hold=None, **kwargs)`. Sie bekommt in `X` ein Bild als Pixelmatrix übergeben. Die anderen Parameter werden einfach an

`pyplot.imshow()` weitergegeben. Die Parameter `extent` und `origin` werden automatisch so gesetzt, das das Bild über die ganze Karte gemalt wird.

Mit der Funktion `warpimage(image='bluemarble', scale=None, **kwargs)` kann man ein Hintergrundbild laden. Dieses Bild muss allerdings den ganzen Globus abdecken. Im Parameter `image` kann man einen Filenamen oder eine URL angeben. Sollte eine URL angegeben sein wird das Bild von der entsprechenden Seite als temporäre Datei herunter geladen. Dabei muss die URL mit `http` anfangen. Sollte nichts angegeben sein wird ein `blue marble next generation` Bild von `http://visibleearth.nasa.gov/` genommen.

Mit der Funktion `wmsimage(server, xpixels=400, ypixels=None, format='png', verbose=False, **kwargs)` kann man ein Hintergrundbild von einem WMS Server laden und zeichnen. Damit diese Funktion funktioniert muss die Karte mit dem passenden Parameter `epsg` erstellt worden sein, oder die Projektion `cyl` gewählt sein.

Mit der Funktion `arcgisimage(server='http://server.arcgisonline.com/ArcGIS', service='ESRI_Imagery_World_2D', xpixels=400, ypixels=None, dpi=96, verbose=False, **kwargs)` kann man ein Hintergrundbild von einem ArcGIS Server laden und darstellen. Mit dem Parameter `service` kann man einstellen welcher Art das Bild sein soll. Um diese Funktion benutzen zu können muss beim erstellen der Karte der Parameter `epsg` passend gesetzt worden sein, oder die Projektion `cyl` gewählt sein. Mit den Parametern `xpixels` und `ypixels` kann man die Anzahl Pixel in der Breite und Höhe einstellen. Sollte `ypixels` nicht gesetzt sein wird die Anzahl Pixel in der Höhe von der Anzahl Pixel in der Breite berechnet, so dass das Seitenverhältnis beibehalten bleibt.

Mit der Funktion `bluemarble(ax=None, scale=None, **kwargs)` kann man ein `blue marbel` Bild als Hintergrund malen. Mit Hilfe des Parameters `scale` kann man die Pixel Anzahl reduzieren, damit das Bild schneller geladen werden kann. Die Standardgröße des Bildes ist 5400x2700.

```
1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n                 llcrnrlon=4.750,urcrnrlon=18.00)
6 m.drawcoastlines()
7 m.bluemarble()
8 #m.fillcontinents(color='coral', lake_color='aqua')
9 # draw parallels and meridians.
10 m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
11 False])
12 m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
13 True, True])
14 #m.drawmapboundary(fill_color='aqua')
15 m.drawrivers(color='aqua')

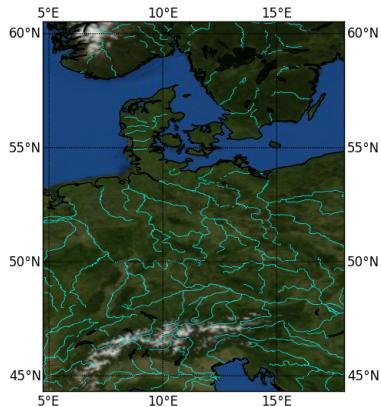
16 #plt.title("Equidistant Cylindrical Projection")
```

```

17 #plt.show()
    plt.savefig( '/home/s.von.hall/seminar/bspmarble.png' )

```

/Users/student/seminar/bsp/bspmarble.py



4.6 Daten plotten

Das Module Basemap bietet die Möglichkeit die Plotroutinen von pyplot zu benutzen, da es die Karte mit matplotlib erstellt. Die Karte ist im Grunde ein `AxesImage`, auf dem dann die entsprechenden Operationen ausgeführt werden. Daher kann man auch die Plotroutinen von pyplot aufrufen da diese ebenfalls mit `AxesImage`s arbeiten. Wenn man mit pyplot zeichnet muss man beachten, dass die Koordinaten, die man angeben muss, in den Projektionskoordinaten der Karte anzugeben sind. Dabei ist die `Basemap` Klasse sehr hilfreich, sie rechnet nämlich Koordinaten in Längen- und Breitengrad in die Projektionskoordinaten um. Dazu mus man einfach die `basemap` Instanz mit den Koordinaten als Parameter aufrufen. Dabei können auch mehrere Koordinaten übergeben werden, indem man zwei Felder übergibt.

```

1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n
6             llcrnrlon=4.750,urcrnrlon=18.00)
7 m.drawcoastlines()
8 m.fillcontinents(color='coral', lake_color='aqua')
# draw parallels and meridians.
9 m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
10 False])
m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
11 True, True])
11 m.drawmapboundary(fill_color='aqua')

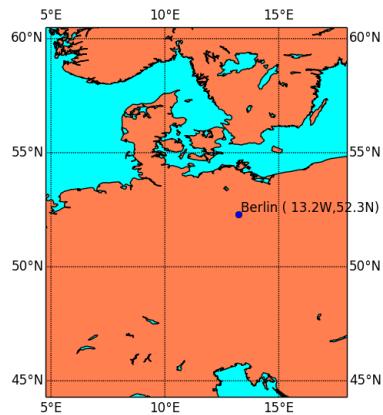
```

```

# plt.title("Equidistant Cylindrical Projection")
13 #plt.show()
lon, lat = 13.2437, 52.3127 # Location of Berlin
15 # convert to map projection coords.
# Note that lon,lat can be scalars, lists or numpy arrays.
17 xpt,ypt = m(lon,lat)
# convert back to lat/lon
19 lonpt, latpt = m(xpt,ypt,inverse=True)
m.plot(xpt,ypt,'bo') # plot a blue dot there
21 # put some text next to the dot, offset a little bit
# (the offset is in map projection coordinates)
23 plt.text(xpt+0.1,ypt+0.1,'Berlin (%5.1fW,%3.1fN)' % (lonpt,latpt
    ))
plt.savefig('/home/s.von.hall/seminar/bspconvert.png')

```

/Users/student/seminar/bsp/bspconvert.py

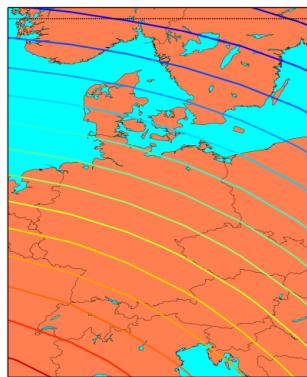


4.6.1 Isobaren plotten

um Isobaren zu plotten braucht man ein zweidimensionales Datenfeld mit den Werten und zwei Felder mit den dazu gehörigen Koordinaten. Diese kann man dann einfach mit Hilfe der Funktion `contour(x, y, data, *args, **kwargs)` plotten. Die Funktion `contour` zeichnet Isobarenlinien. Mit der Funktion `contourf(x, y, data, *args, **kwargs)` werden diese auch gefüllt. Den beiden Funktionen kann man ein `colormap` Objekt im Parameter `cmap` übergeben. Dieser Parameter wird dann einfach an `pyplot` weitergegeben. Darüber kann man den Farbverlauf der Isobaren steuern. Über den Parameter `levels` kann man angeben welche Wertelevel gezeichnet werden sollen.

```
from mpl_toolkits.basemap import Basemap
1 import matplotlib.pyplot as plt
2 import numpy as np
# set up orthographic map projection with
# perspective of satellite looking down at 50N, 100W.
3 # use low resolution coastlines.
4 map = Basemap(projection='cyl',resolution='i', llcrnrlat=44.30,
5     urcrnrlat=60.5,\n
6         llcrnrlon=4.750,urcrnrlon=18.00)# draw coastlines,
7         country_boundaries, fill_continents.
8 map.drawcoastlines(linewidth=0.25)
9 map.drawcountries(linewidth=0.25)
10 map.fillcontinents(color='coral',lake_color='aqua')
11 # draw the edge of the map projection region (the projection
12 # limb)
13 map.drawmapboundary(fill_color='aqua')
14 # draw lat/lon grid lines every 30 degrees.
15 map.drawmeridians(np.arange(0,360,30))
16 map.drawparallels(np.arange(-90,90,30))
17 # make up some data on a regular lat/lon grid.
18 nlats = 73; nlons = 145; delta = 2.*np.pi/(nlons-1)
19 lats = (0.5*np.pi-delta*np.indices((nlats,nlons))[0,:,:])
20 lons = (delta*np.indices((nlats,nlons))[1,:,:])
21 wave = 0.75*(np.sin(2.*lats)**8*np.cos(4.*lons))
22 mean = 0.5*np.cos(2.*lats)*((np.sin(2.*lats))**2 + 2.)
23 # compute native map projection coordinates of lat/lon grid.
24 x, y = map(lons*180./np.pi, lats*180./np.pi)
25 # contour data over the map.
26 cs = map.contour(x,y,wave+mean,15,linewidths=1.5)
27 #plt.title('contour lines over filled continent background')
28 #plt.show()
29 plt.savefig('/home/s.von.hall/seminar/bspcont.png')
```

/Users/student/seminar/bsp/bspcontur.py



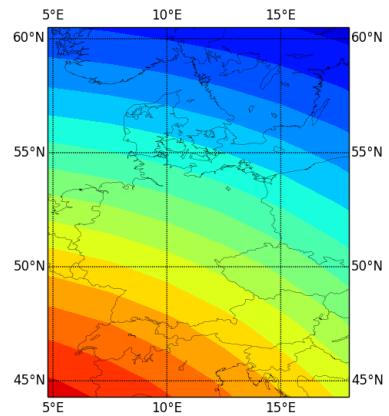
Mit der Funktion `contourf()`

muss man aufpassen, dass die Graphik nicht vom Hintergrund übermalt wird. Daher habe ich in dem Beispiel den Kontinent nicht gefüllt.

```

1 from mpl_toolkits.basemap import Basemap
2 import matplotlib.pyplot as plt
3 import numpy as np
4 # set up orthographic map projection with
5 # perspective of satellite looking down at 50N, 100W.
6 # use low resolution coastlines.
7 map = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
8               urcrnrlat=60.5,\ 
9               llcrnrlon=4.750,urcrnrlon=18.00)# draw coastlines ,
10              country_boundaries , fill_continents .
11 map.drawcoastlines(linewidth=0.25)
12 map.drawcountries(linewidth=0.25)
13 #map.fillcontinents(color='coral', lake_color='aqua')
14 # draw the edge of the map projection region (the projection
15 # limb)
16 map.drawmapboundary(fill_color='aqua')
17 # draw lat/lon grid lines every 30 degrees.
18 map.drawparallels(np.arange(-90.,91.,5), labels=[True, True,
19             True, False])
20 map.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
21             True, True])
22 # make up some data on a regular lat/lon grid.
23 nlats = 73; nlons = 145; delta = 2.*np.pi/(nlons-1)
24 lats = (0.5*np.pi-delta*np.indices((nlats,nlons))[0,:,:])
25 lons = (delta*np.indices((nlats,nlons))[1,:,:])
26 wave = 0.75*(np.sin(2.*lats)**8*np.cos(4.*lons))
27 mean = 0.5*np.cos(2.*lats)*((np.sin(2.*lats))**2 + 2.)
28 # compute native map projection coordinates of lat/lon grid.
29 x, y = map(lons*180./np.pi, lats*180./np.pi)
30 # contour data over the map.
31 cs = map.contourf(x,y, wave+mean, 15, linewidths=1.5)
32 #plt.title('contour lines over filled continent background')
33 #plt.show()
34 plt.savefig('/home/s.von.hall/seminar/bspcontf.png')
```

```
/Users/student/seminar/bsp/bspconturf.py
```

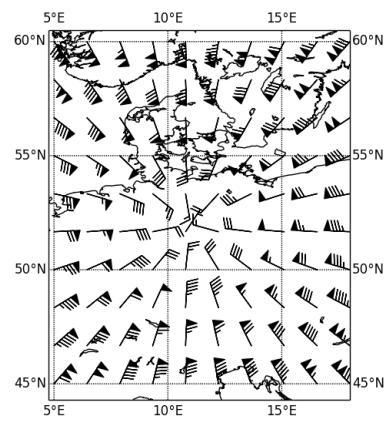


4.6.2 Windmarken plotten

Mit der Funktion `barbs(**kw)` kann man einfach Windmarken plotten. Dazu übergibt man der Funktion die Koordinaten der Marke und die Koordinaten des Endpunkts des Vektors der dargestellt werden soll. Die Form der Marke wird durch die Länge des Vektors bestimmt. Eine Flagge bedeutet einen Wert von 50, ein voller Balken 10, ein halber Balken 5. Mit dem Parameter `barb_increments` kann man eigene Werte festlegen. Dafür wird dem Parameter ein `dictionary` mit den Schlüsseln `half`, `full`, `flag` übergeben. Mit dem Parameter `length` kann man die Länge der Fähnchen in Punkten angeben, der Rest der Marke wird dagegen skaliert.

```
1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n             llcrnrlon=4.750,urcrnrlon=18.00)
6 m.drawcoastlines()
7 #m.fillcontinents(color='coral', lake_color='aqua')
8 # draw parallels and meridians.
9 m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
10                 False])
11 m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
12                 True, True])
13 #m.drawmapboundary(fill_color='aqua')
14 x = np.linspace(5, 18, 10)
15 y=np.linspace(45, 60, 10)
16 X,Y = np.meshgrid(x, y)
17 U, V = 12*(X-11), 12*(Y-52)
18 #Default parameters, uniform grid
19 m.barbs(X, Y, U, V)
20 #plt.show()
21 plt.savefig('/home/s.von.hall/seminar/bspbarbs.png')
```

/Users/student/seminar/bsp/bspbarbs.py

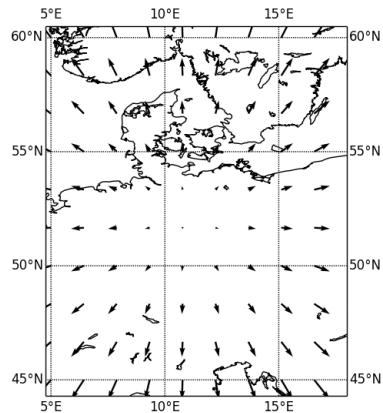


4.6.3 Windvektoren plotten

Mit der Funktion `quiver(**kw)` kann man Vektoren plotten. Hierbei werden wieder 2 Koordinaten angegeben, wie bei dem Plotten von Windmarken. Mit den Parametern `u,v` wird ein Vektor übergeben der gezeichnet werden soll. Mit den Parametern `scale`, `scale_units` kann man bestimmen wie lang die Vektoren werden. Je kleiner der `scale` Parameter desto größer wird der Vektor. Wenn man bei diesen Parametern nichts angibt wird der Wert automatisch aus den zu zeichnenden Vektoren ermittelt.

```
1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n      llcrnrlon=4.750,urcrnrlon=18.00)
6 m.drawcoastlines()
7 #m.fillcontinents(color='coral', lake_color='aqua')
8 # draw parallels and meridians.
9 m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
10                 False])
11 m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
12                 True, True])
13 #m.drawmapboundary(fill_color='aqua')
14 x = np.linspace(5, 18, 10)
15 y=np.linspace(45, 60, 10)
16 X,Y = np.meshgrid(x, y)
17 U, V = 12*(X-11), 12*(Y-52)
18 #Default parameters, uniform grid
19 m.quiver(X, Y, U, V)
20 #plt.show()
21 plt.savefig('/home/s.von.hall/seminar/bspquiver.png')
```

/Users/student/seminar/bsp/bspquiver.py

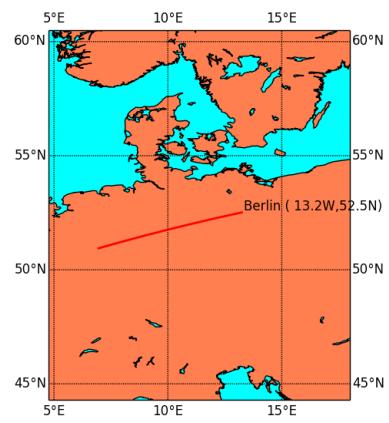


4.6.4 gekrümmte Strecken plotten

Um Strecken wie zum Beispiel eine Flugroute von New York nach London zu plotten, benutzt man am Einfachsten die Funktion `drawgreatcircle(lon1, lat1, lon2, lat2, del_s=100.0, **kwargs)`. Diese Funktion zeichnet einen Bogen von den Koordinaten `lon1, lat1` nach `lon2, lat2` dabei werden Punkte alle `del_s` km ermittelt. Diese Punkte kann man sich auch mit der Funktion `gcpoints(lon1, lat1, lon2, lat2, points)` berechnen lassen.

```
1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\n                 llcrnrlon=4.750,urcrnrlon=18.00)
6 m.drawcoastlines()
7 m.fillcontinents(color='coral', lake_color='aqua')
# draw parallels and meridians.
8 m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
      False])
9 m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
      True, True])
10 m.drawmapboundary(fill_color='aqua')
# plt.title("Equidistant Cylindrical Projection")
11 #plt.show()
12 lak, lok = 50.941357, 6.958307 # Lokation von Koeln
13 lon, lat = 13.2437, 52.5127 # Location of Berlin
# convert to map projection coords.
14 # Note that lon,lat can be scalars, lists or numpy arrays.
15 xpt,ypt = m(lon,lat)
# convert back to lat/lon
16 lonpt, latpt = m(xpt,ypt,inverse=True)
17 m.drawgreatcircle(xpt,ypt,lok,lak,color='r', linewidth=2) #
      plot a blue dot there
# put some text next to the dot, offset a little bit
18 # (the offset is in map projection coordinates)
19 plt.text(xpt+0.1,ypt+0.1,'Berlin (%5.1fW,%3.1fN)' % (lonpt, latpt
      ))
20 plt.savefig('/home/s.von.hall/seminar/bspgreatcirc.png')
```

/Users/student/seminar/bsp/bspgreatcirc.py



4.6.5 Viele Punkte plotten

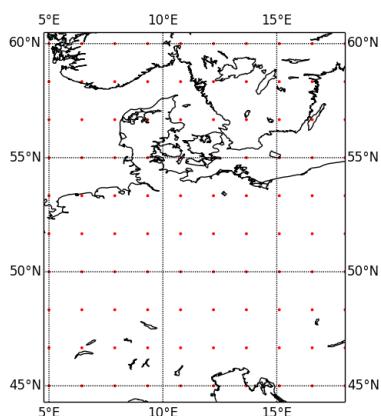
Mit der Funktion `scatter(x, y, s=20, c=u'b', marker=u'o', cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, verts=None, hold=None, **kwargs)` kann man Marken plotten. Diese werden an die Positionen die durch `x,y` definiert sind gezeichnet. Der Parameter `s` gibt die Größe der Marke an. Der Parameter `marker` gibt an was für eine Marke gezeichnet werden soll. Über den Parameter `c` kann man die Farben der Marken bestimmen.

```

1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib.pyplot as plt
4 m = Basemap(projection='cyl', resolution='i', llcrnrlat=44.30,
5             urcrnrlat=60.5,\ 
6                 llcrnrlon=4.750,urcrnrlon=18.00)
7 m.drawcoastlines()
8 #m.fillcontinents(color='coral', lake_color='aqua')
# draw parallels and meridians.
9 m.drawparallels(np.arange(-90.,91.,5), labels=[True, True, True,
10               False])
11 m.drawmeridians(np.arange(-180.,181.,5), labels=[True, False,
12               True, True])
13 #m.drawmapboundary(fill_color='aqua')
14 x = np.linspace(5, 18, 10)
15 y=np.linspace(45, 60, 10)
16 X,Y = np.meshgrid(x, y)
17 U, V = 12*(X-11), 12*(Y-52)
#Default parameters, uniform grid
18 m.scatter(X, Y, 3, marker='o', color='r')
19 #plt.show()
20 plt.savefig('/home/s.von.hall/seminar/bspscatter.png')

```

/Users/student/seminar/bsp/bspscatter.py



4.6.6 Linienzüge zeichnen

Linienzüge lassen sich mit der Funktion `plot(x,y,**kw)` zeichnen. Diese Funktion zeichnet einen Linie von einem Punkt zum Nächsten. Dabei kann man `x,y` einfach als Felder übergeben. Hierbei ist zu beachten das davon ausgegangen wird das die Koordinaten Projektionskoordinaten sind.

4.7 Karte speichern

Um die Karte die man erstellt hat zu speichern bedient man sich des Moduls `pyplot`. Von diesem Modul kann man die Funktion `savefig(fname, dpi=None, facecolor='w', edgecolor='w', orientation='portrait', papertype=None, format=None, transparent=False, bbox_inches=None, pad_inches=0.1, frameon=None)` benutzen um die Figur zu speichern. Dazu gehören auch die Farblegende und der Titel der Figur. Es wird also nicht nur die Karte gespeichert. Mit dem Parameter `fname` wird der Dateiname angegeben in den gespeichert werden soll. Mit dem Parameter `format` kann ein unterstütztes Format angegeben werden, diese können variieren, meistens werden die Formate `png`, `pdf`, `ps`, `eps`, `svg` unterstützt. Die Parameter `papertype` und `orientation` werden eventuell nur für das `ps` Format unterstützt.

4.8 Karte anzeigen

Mit der `pyplot` Funktion `show()` kann man die Karte anzeigen lassen. In der Anzeige kann man dann auch zoomen und die Karte verschieben. Was es einem ermöglicht auch Zeichnungen außerhalb der eigentlichen Karte zu sehen.

Literatur

<http://matplotlib.org/basemap> Basemap Dokumentation.

http://resources.arcgis.com/en/help/main/10.1/index.html#/List_of_supported_map_projections-003r0000001700000/ ArcGIS Kartendefinitionen

<http://matplotlib.org/index.html> Die matplotlib Dokumentation

http://en.wikipedia.org/wiki/Category:Cartographic_projections Wikipedia
Kategorie Kartenprojektionen

Nomenklatur

\mathcal{R}

Der Radius der Erde.

φ

Der Breitengrad der Polarkoordinate.

λ

Der Längengrad der Polarkoordinate.

x

Die X Koordinate der 2D Projektion.

y

Die Y Koordinate der 2D Projektion.

Glossar

Flächentreu

Flächentreu heißt, dass der Maßstab mit dem Flächen verkleinert werden auf der gesamten Karte gleich ist. Dies führt an den Rändern zu Verzerrungen.

Winkeltreu

Winkeltreu heißt, dass die Winkel beziehungsweise die Richtungen von geraden beibehalten bleiben.

Längentreu

Längentreu heißt, dass die Entferungen zwischen Punkten mit dem gleichen Maßstab verkleinert werden.

5 Anhang

5.1 Azimuthale äquidistante Projektion

Bei dieser Projektion ist die kürzeste Entfernung vom Mittelpunkt der Karte zu einem beliebigen Anderen Punkt eine gerade Linie. Das bedeutet das alle Punkte die in auf einem Kreis um den Kartenmittelpunkt liegen, äquidistant sind. Nachteil:

Die Gebiete die auf der anderen Seite der Welt liegen werden sehr verzehrt dargestellt. Daher ist diese Projektion für Weltkarten eher ungeeignet.

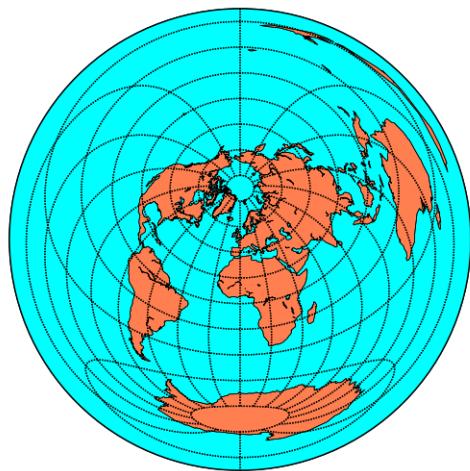


Abbildung 1: Azimuthale äquidistante Projektion

5.2 Gnomonische Projektion

In der gnomonischen Projektion werden alle Längenkreise als gerade Linien dargestellt. Das Besondere der gnomonischen Projektion ist das der Projektionspunkt im Mittelpunkt der Erde liegt. Da hier von Innen nach Außen projiziert wird, nimmt die Verzerrung mit der Entfernung vom Kartenmittelpunkt zu. Die gnomonische Projektion ist keine globale Projektion.

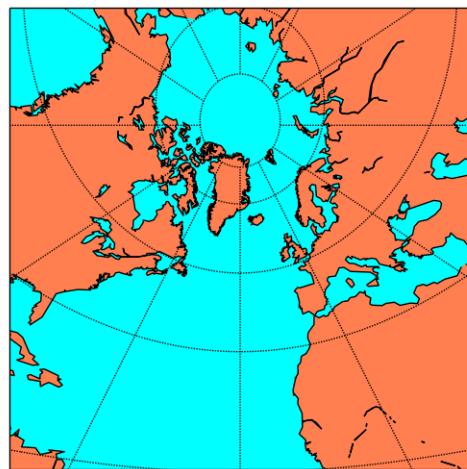


Abbildung 2: Gnomonische Projektion

5.3 Orthographische Projektion

Bei der orthographischen Projektion wird die Erde aus einer unendlichen Entfernung abgebildet. Die Verzerrung ist an den Grenzen nicht übermäßig stark. Die orthographische Projektion ist keine globale Projektion. Durch die große Entfernung wirkt die Projektion dreidimensional.

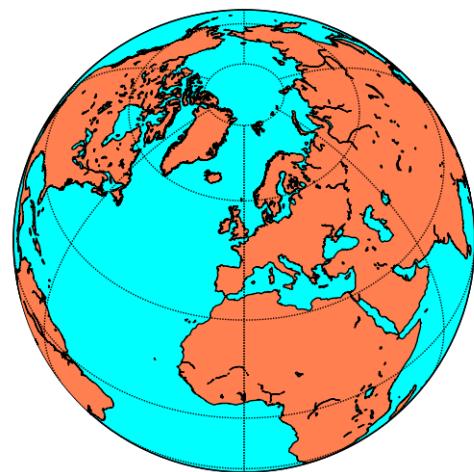


Abbildung 3: Orthographische Projektion

5.4 Geostationäre Projektion

In der geostationären Projektion wird die Erde aus der Perspektive eines geostationären Satelliten. Vorteil:

- Wenn die Position des Satelliten bekannt ist, kann man dessen Bilder als Hintergrund verwenden (siehe 4.5)

Nachteil:

- Die andere Seite der Erde wird nicht dargestellt.
- Entfernungen zwischen 2 Punkten werden auf Kreisbögen gemessen.
- Der Satellit muss über dem Äquator sein.

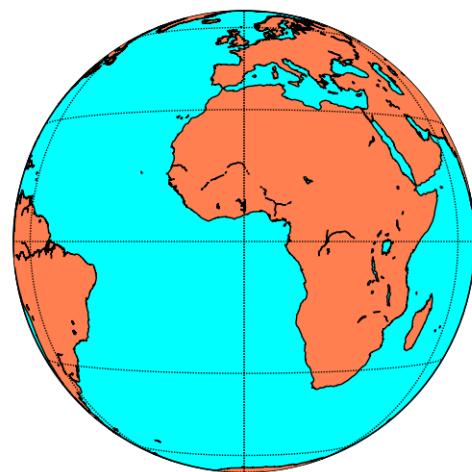


Abbildung 4: Geostationäre Projektion

5.5 Nah-seitige perspektivische Projektion

Die Nah-seitige Perspektive zeigt die Erde aus der Sicht eines Satelliten. Also ist es im Prinzip das selbe wie die geostationäre Projektion. Allerdings muss der Satellit nicht über dem Äquator sein.

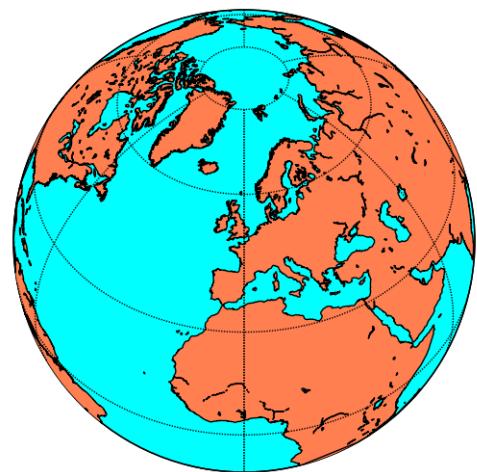


Abbildung 5: Nah-seitige perspektivische Projektion

5.6 Mollweide Projektion

Bei der Mollweiden Projektion wird die Erde als Oval dargestellt. Die Mollweiden Projektion ist flächentreu. Der Äquator und der Nullmeridian werden bei der Mollweiden Projektion maßstabsgetreu wieder gegeben. Breitenkreise werden bei der Mollweiden Projektion als Geraden dargestellt. Die Längenkreise sind als Ellipsen dargestellt.

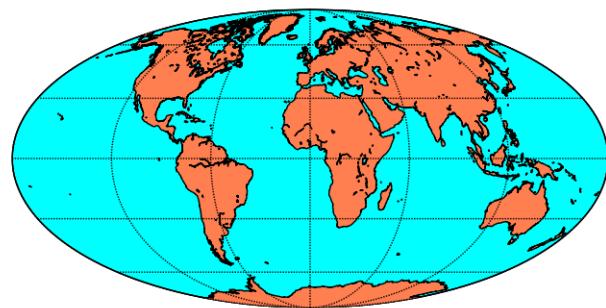


Abbildung 6: Mollweiden Projektion

5.7 Hammer Projektion

Die Hammer Projektion ist wie die Mollweiden Projektion eine flächentreue Projektion. Bei der Hammer Projektion wird die Erden ebenfalls als Oval dargestellt. Allerdings werden die Breitenkreise im Gegensatz zur Mollweiden Projektion als Ellipsen dargestellt, dadurch ist die Verzerrung an den Rändern nicht so stark. Nachteil bei dieser Art der Darstellung ist, dass die Erde an den Polen gestaucht wird.

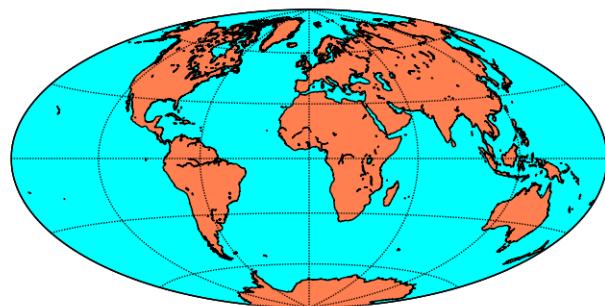


Abbildung 7: Hammer Projektion

5.8 Robinson Projektion

Die Robinson Projektion ist eine globale Projektion. Die Erde wird hier annähernd Oval dargestellt. Die Pole werden allerdings in dieser Darstellung nicht abgedeckt. Breitenkreise werden in der Robinson Projektion als Geraden dargestellt. Bei dieser Darstellung wurden die Verzerrungen reduziert. Nachteil der Robinson Projektion ist das die Pole nicht erfasst werden.

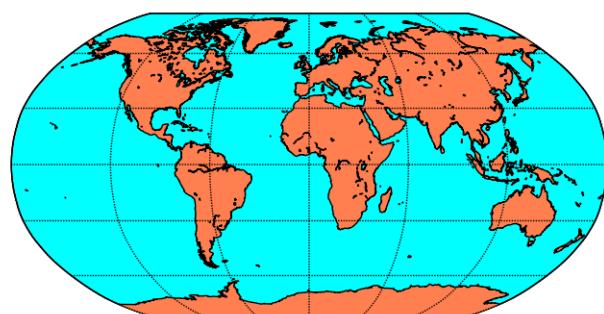


Abbildung 8: Robinson Projektion

5.9 Eckert 4 Projektion

Die Eckert 4 Projektion ist sehr ähnlich wie die Robinson Projektion, allerdings ist Sie flächentreu. Deshalb ist die Darstellung an den Polen gestaucht. Die Erde wird wie auf einem Reifen dargestellt. Die Seitenränder sind in dieser Projektion Halbkreise.

Formel:

$$\begin{aligned}\mathcal{X} &= \frac{2}{\sqrt{4\pi + \pi^2}} \mathcal{R}(\lambda - \lambda_0) (1 + \cos \theta) \\ \mathcal{Y} &= 2\sqrt{\frac{\pi}{4 + \pi}} \mathcal{R} \sin \theta \\ \theta + \sin \theta \cos \theta + 2 \sin \theta &= \left(2 + \frac{\pi}{2}\right) \sin \varphi\end{aligned}$$

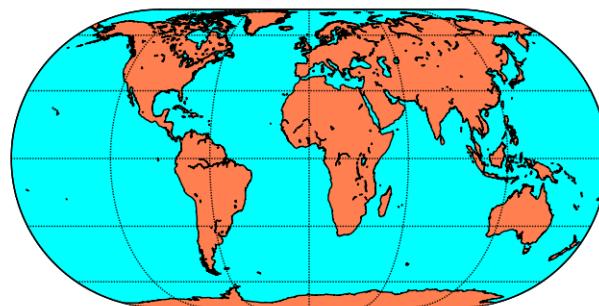


Abbildung 9: Eckert 4 Projektion

5.10 Kavrayskiy 7 Projektion

Die Kavrayskiy 7 Projektion ist der Robinson Projektion sehr ähnlich. Sie stellt die Erde wieder annähernd oval dar. Die Breitenkreise werden in dieser Projektion als Geraden dargestellt. Diese Projektion stellt einen Kompromiss zwischen winkeltreuen und flächentreuen Projektionen dar. Die Pole sind in dieser Darstellung sehr breit gezogen.

Formel:

$$\begin{aligned}x &= \frac{3\lambda}{2\pi\sqrt{\frac{\pi^2}{3} - \varphi^2}} \\y &= \varphi\end{aligned}$$

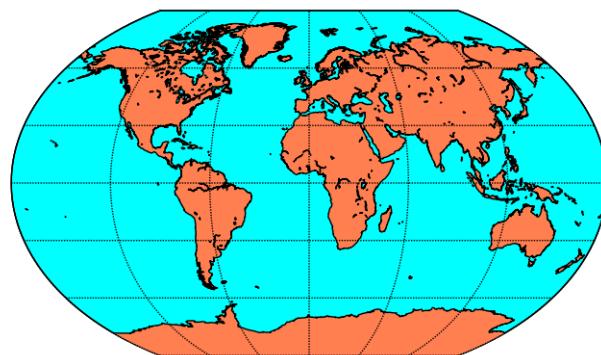


Abbildung 10: Kavrayskiy 7 Projektion

5.11 McBryde-Thomas Projektion

Diese Projektion ist eine flächentreue Darstellung der Erde. Die Breitenkreise werden als Geraden dargestellt. Die Längenkreise werden als Bögen dargestellt. Dabei haben die Längenkreise, auf einem Breitenkreis, immer den gleichen Abstand zueinander. Die Breitenkreise hingegen stehen immer näher je näher man den Polen kommt. Die Pole sind auf ein drittel des Äquators gestreckt. Der Nullmeridian ist in dieser Projektion 0,45 mal so lang wie der Äquator.

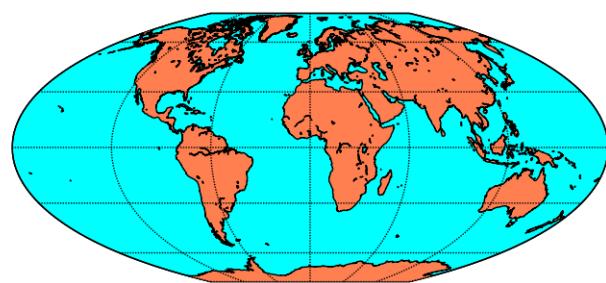


Abbildung 11: McBryde-Thomas Projektion

5.12 Sinus-förmige Projektion

Die Sinus-förmige Projektion ist eine flächentreue Projektion. Auch in dieser Projektion werden die Breitenkreise als Geraden dargestellt. Das besondere an der Sinus-förmigen Projektion ist das die Länge der Breitenkreise relational zu $\cos \varphi$ ist, dies führt zu einer starken Verzerrung außerhalb der Mitte. Vorteil der Sinus-förmigen Projektion:

- Die Projektion ist einfach zu berechnen.

Nachteil der Sinus-förmigen Projektion:

- Die Projektion ist nicht sehr anschaulich.

Formel:

$$\begin{aligned}x &= R \cdot (\lambda - \lambda_0) \cdot \cos \varphi \\y &= R \cdot \varphi\end{aligned}$$

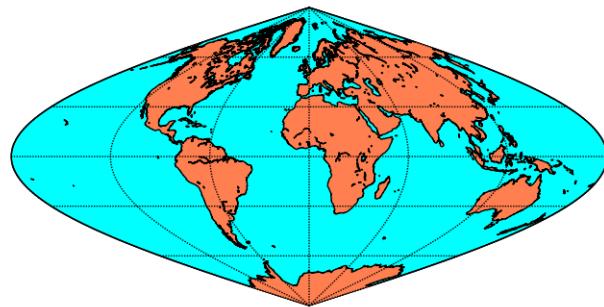


Abbildung 12: Sinus-förmige Projektion

5.13 Äquidistante Zylinder Projektion

Die äquidistante Zylinder Projektion ist die einfachste Projektion. Sie stellt die Erde einfach in Längen- und Breitengrad dar. Dabei entsteht ein gleichmäßiges Gitterraster. Die Projektion ist weder winkel- noch flächentreu, das heißt, dass die Verzerrung mit der Entfernung vom Mittelpunkt der Karte zunimmt.

Vorteil der äquidistanten Zylinder Projektion:

- Die Projektion ist sehr einfach zu berechnen.

Nachteil der äquidistanten Zylinder Projektion:

- Die Verzerrungen wirken sowohl auf die Fläche als auch auf die Abstände aus.

Formel:

$$\begin{aligned}x &= \lambda \\y &= \varphi\end{aligned}$$

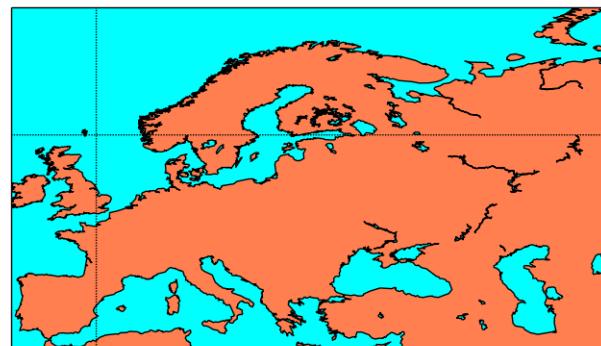


Abbildung 13: Äquidistante Zylinderprojektion

5.14 Cassini Projektion

Bei der Cassini Projektion wird die Erde zuerst gedreht, sodass der Äquator zum Nullmeridian wird. Danach wird dann eine äquidistante Zylinder Projektion angewendet.

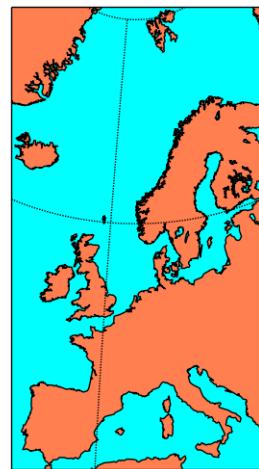


Abbildung 14: Cassini Projektion

5.15 Mercatorprojektion

Die Mercatorprojektion ist eine winkeltreue Zylinderprojektion. Sie erreicht dabei nie Pole. Die Längengrade verlaufen in dieser Projektion parallel und haben den gleichen Abstand zueinander. Die Breitengrade sind ebenfalls parallel zueinander haben aber unterschiedliche Abstände. Die Verzerrung nimmt mit Polnähe zu, das heißt, dass die Karte zu den Polen hin gestreckt wird.

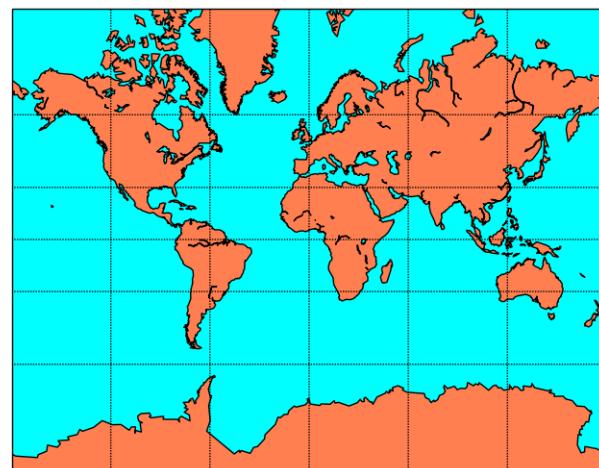


Abbildung 15: Mercatorprojektion

5.16 Transversale Mercatorprojektion

Bei der transversalen Mercatorprojektion wird der Globus zuerst um 90° gedreht, so dass der 0 Meridian zu Äquator wird. Danach wird eine normale Mercatorprojektion erstellt.

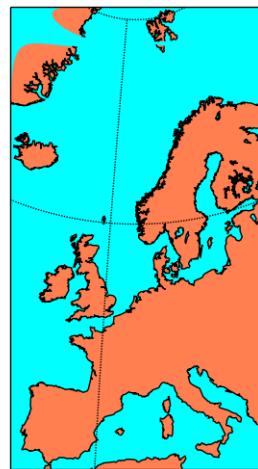


Abbildung 16: Transversale Mercatorprojektion

5.17 Schiefe Mercatorprojektion

Bei der schiefen Mercatorprojektion kann die zentrale Linie jede Linie zwischen zwei Punkten sein und nicht wie bei der Mercatorprojektion nur ein Breitengrad oder der transversen Mercatorprojektion nur ein Längengrad.

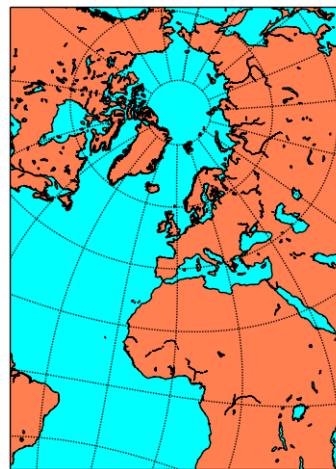


Abbildung 17: Schiefe Mercatorprojektion

5.18 Polykonische Projektion

Die polikonische Projektion ist eine globale Projektion. Hierbei werden auf dem zentralen Meridian unendlich viele Kegel aufgebaut. Dabei entstehen nicht konzentrische Breitenkreise. Die Projektion geht an den Polen auseinander. Die Verzerrung der Fläche nimmt mit der Entfernung vom zentralen Meridian der Karte zu. Die Winkel sind lokal entlang des zentralen Längengrades genau, sonst sind sie verzerrt.

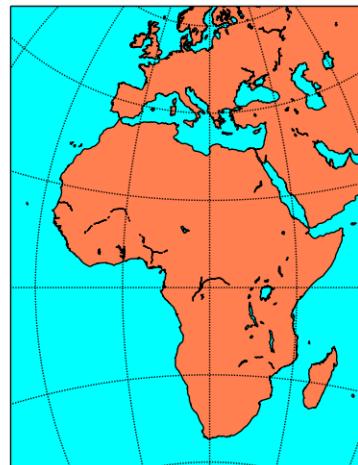


Abbildung 18: Polykonische Projektion

5.19 Miller Zylinderprojektion

Die Miller Zylinderprojektion ist eine globale Projektion. Sie ist der Mercatorprojektion sehr ähnlich. Allerdings ist hier die Verzerrung an den Polen anders. Die Pole werden nicht mehr so stark gesteckt, dafür ist die Miller-Projektion nicht winkeltreu. Dafür reicht die Miller Zylinderprojektion bis an die Pole.

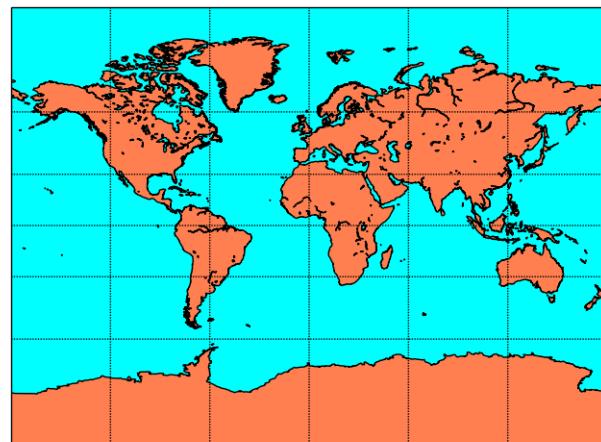


Abbildung 19: Miller Zylinderprojektion

5.20 Stereographische Gall Projektion

Die stereographische Gall Projektion ist eine globale Zylinderprojektion. Die Gall Projektion hat zwei Standartparallelen bei 45°N und 45°S . Die Verzerrung der Fläche und Winkel nimmt mit Abstand zu den Standartparallelen zu. Die Verzerrung ist allgemein an den Polen sehr stark.

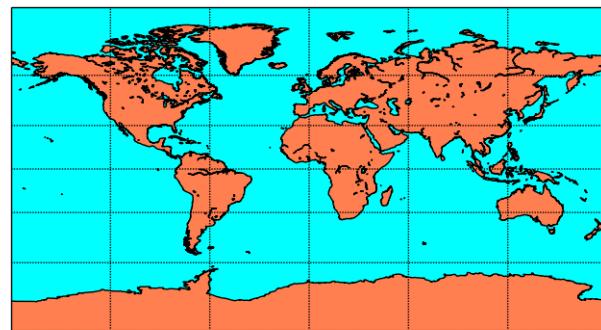


Abbildung 20: Stereographische Gall Projektion

5.21 Flächentreue Zylinderprojektion

Die flächentreue Zylinderprojektion ist eine flächentreue Zylinderprojektion wie es der Name bereits sagt.

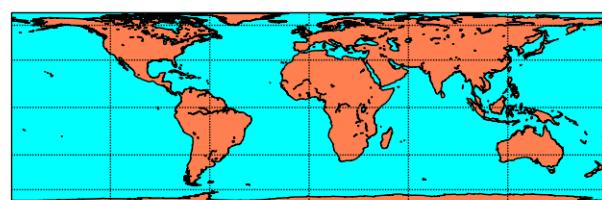


Abbildung 21: Flächentreue Zylinderprojektion

5.22 Winkeltreue Lambert Projektion

Die winkeltreue Lambert Projektion ist winkeltreu. Die Längenkreise werden als Geraden dargestellt. Die winkeltreue Lambert Projektion ist eine Kegelprojektion.

Formel:

$$\begin{aligned}\mathcal{X} &= \rho \sin(n(\lambda - \lambda_0)) \\ \mathcal{Y} &= \rho_0 - \rho \cos(n(\lambda - \lambda_0)) \\ \rho &= F \cot^n\left(\frac{1}{4}\pi + \frac{1}{2}\varphi\right) \\ n &= \frac{\ln(\cos \varphi_1 \sec \varphi_2)}{\ln(\tan(\frac{1}{4}\pi + \frac{1}{2}\varphi_2) \cot(\frac{1}{4}\pi + \frac{1}{2}\varphi_1))} \\ F &= \frac{\cos \varphi_1 \tan^n(\frac{1}{4}\pi + \frac{1}{2}\varphi_1)}{n}\end{aligned}$$

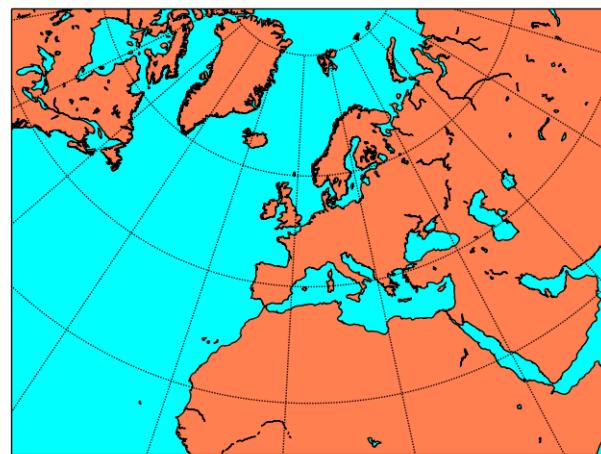


Abbildung 22: Winkeltreue Lambert-Projektion

5.23 Azimuthale Flächentreue Lambert-Projektion

Die flächentreue Lambert-Projektion ist eine globale Projektion. Diese Projektion wird um einen Punkt herum aufgebaut. Dabei bleibt der 3D Abstand jedes Punktes zum Mittelpunkt der Projektion erhalten. Das sorgt dafür das Winkel mit zunehmender Entfernung vom Mittelpunkt stärker verzerrt werden. Diese Projektion stellt die Erde als Kreis dar.

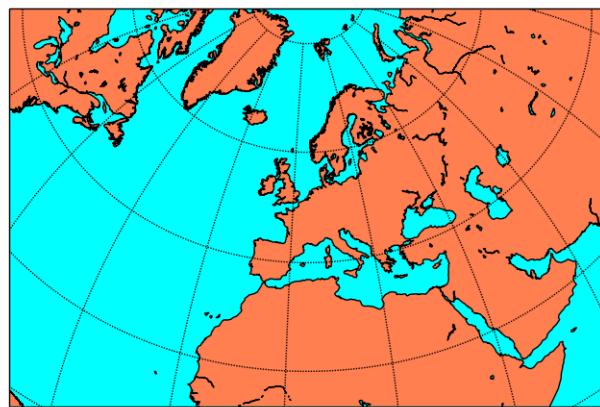


Abbildung 23: Azimuthale flächentreue Lambert-Projektion

5.24 Stereografische Projektion

Die stereografische Projektion ist eine winkeltreue Projektion. Sie ist allerdings nicht flächentreu. Man kann mit dieser Projektion die ganze Erde abbilden allerdings nimmt die Verzerrung sehr schnell stark zu, weshalb die stereografische Projektion für globale Abbildungen eher ungeeignet ist. Diese Projektion erhält man, wenn man von einem Ausgangspunkt geraden durch jeden Punkt der Erde zieht, die Schnittpunkte dieser Geraden mit der Projektionsebene sind die Punkte auf die projiziert wird. Die Projektionsebene kann grundsätzlich frei positioniert werden.

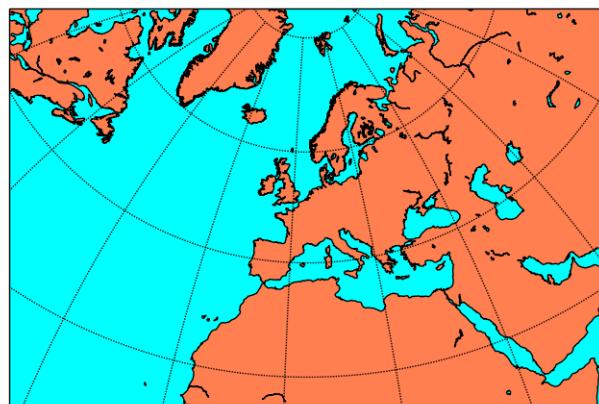


Abbildung 24: Stereografische Projektion

5.25 Längentreue Kegelprojektion

Die längentreue Kegelprojektion ist eine globale Projektion. Sie ist weder winkel- noch flächentreu. Die Verzerrung nimmt zum Kartenrand hin zu. Die Breitenkreise sind in dieser Projektion gleichmäßig verteilt, und die Längenkreise werden als Geraden dargestellt. Die Projektion bildet die Erde auf einen Kegelmantel ab.

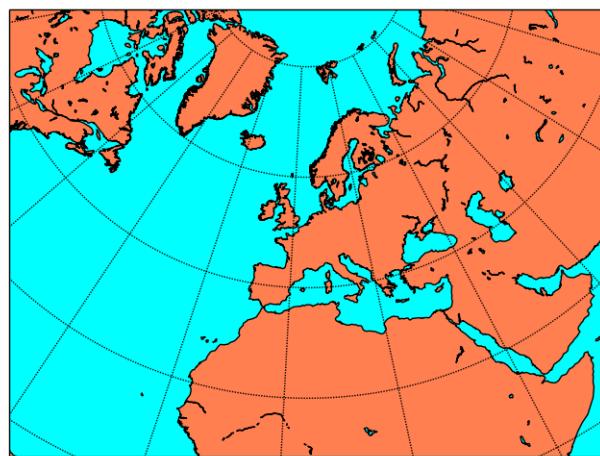


Abbildung 25: Längentreue Kegelprojektion

5.26 Flächentreue Albert-Projektion

Die flächentreue Albert-Projektion ist eine globale Kegelprojektion. Sie bildet die Erde auf einen Kegelmantel ab. Sie benutzt für die Projektion 2 Standardbreitenkreise.

Formel:

$$\begin{aligned}\mathcal{X} &= \rho \sin \theta \\ \mathcal{Y} &= \rho_0 - \rho \cos \theta \\ n &= \frac{1}{2}(\sin \varphi_1 + \sin \varphi_2) \\ \theta &= n(\lambda - \lambda_0) \\ \tau &= \cos^2 \varphi_1 + 2n \sin \varphi_1 \\ \rho &= \frac{\sqrt{\tau - 2n \sin \varphi}}{n} \\ \rho_0 &= \frac{\sqrt{\tau - 2n \sin \varphi_0}}{n}\end{aligned}$$

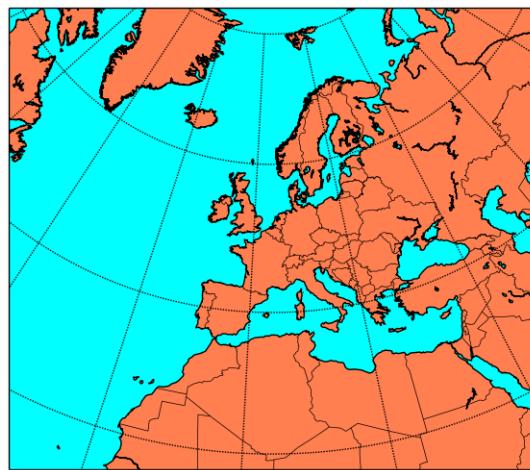


Abbildung 26: Flächentreue Albert-Projektion

5.27 Polare stereographische Projektion

Die polare stereographische Projektion ist eine stereographische Projektion die einen der beiden Pole als Kartenzentrum hat.

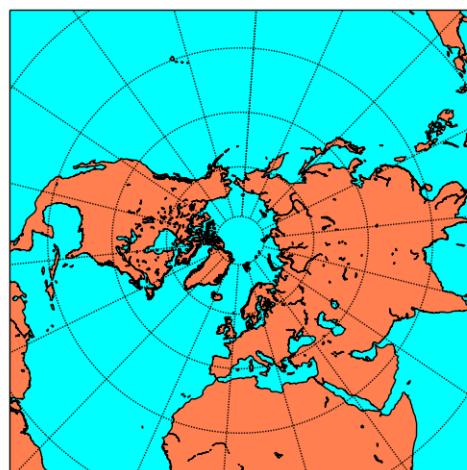


Abbildung 27: Nordpol

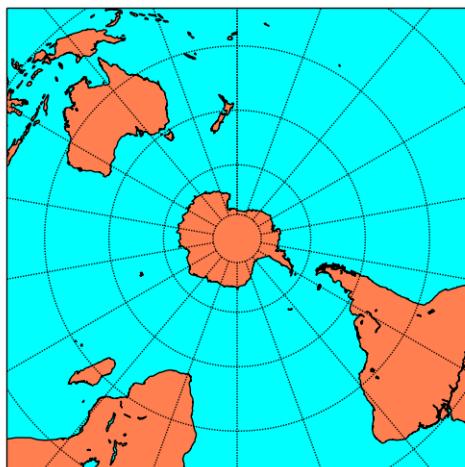


Abbildung 28: Südpol

5.28 Polare azimutale Lambertprojektion

Die polare azimutale Lambertprojektion hat ebenfalls einfach nur einen Pol als Kartenzentrum.

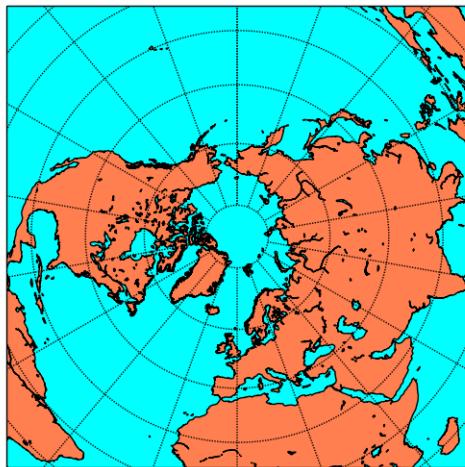


Abbildung 29: Nordpol

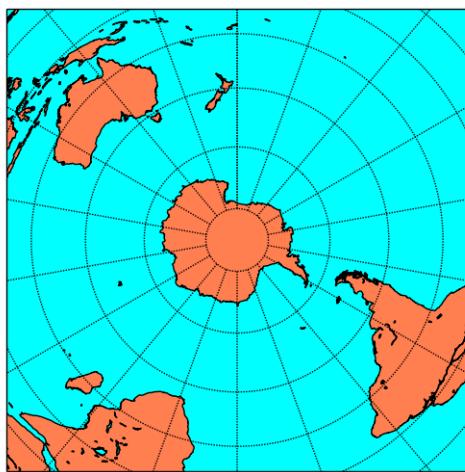


Abbildung 30: Südpol

5.29 Polare azimuthale äquidistante Projektion

Diese Projektion hat ebenfalls einfach einen Pol als Mittelpunkt.

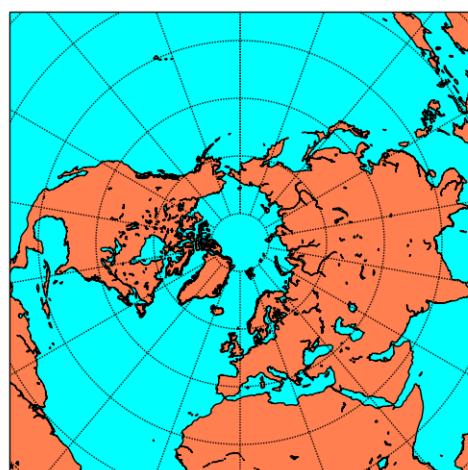


Abbildung 31: Nordpol

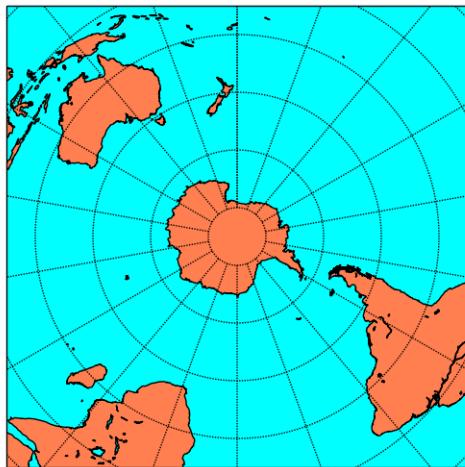


Abbildung 32: Südpol

5.30 Van der Grinten Projektion

Die van der Grinten Projektion ist eine globale Projektion, die die Erde auf einen Kreis projiziert. Diese Projektion ist weder winkel- noch flächentreu. Die Verzerrung nimmt zum Rand hin zu. Die van der Grinten Projektion ist um den Äquator zentriert.

Formel:

$$\begin{aligned}
x &= \frac{\pm\pi(A(G-P^2) + \sqrt{A^2(G-P^2)^2 - (A^2+P^2)(G^2-P^2)})}{P^2+A^2} \\
y &= \frac{\pm\pi(PQ - A\sqrt{(A^2+1)(P^2+A^2)-Q^2})}{P^2+A^2} \\
A &= \frac{1}{2}\left|\frac{\pi}{\lambda-\lambda_0} - \frac{\lambda-\lambda_0}{\pi}\right| \\
G &= \frac{\cos\theta}{\sin\theta+\cos\theta-1} \\
P &= G\left(\frac{2}{\sin\theta}-1\right) \\
\theta &= \arcsin\left|\frac{2\varphi}{\pi}\right| \\
Q &= A^2+G \\
&\text{Falls } \varphi = 0 : \\
x &= \lambda - \lambda_0 \\
y &= 0 \\
&\text{Falls } \lambda = \lambda_0 \text{ oder } \varphi = \pm\frac{\pi}{2} : \\
x &= 0 \\
y &= \pm\pi\tan\frac{\theta}{2}
\end{aligned}$$

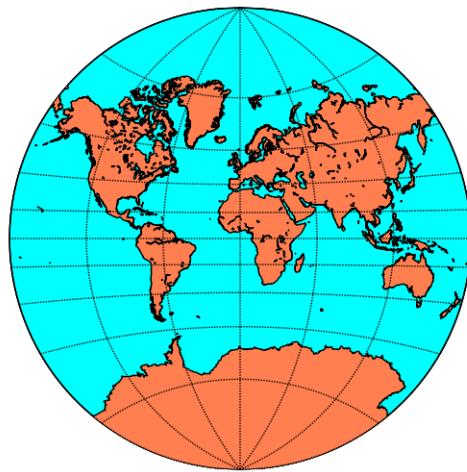


Abbildung 33: Van der Grinten Projektion